UNIVERSIDADE DE ÉVORA

Compressão por Entropia

Relatório Técnico

Tomás Rato nº62755
João Padeiro nº63655

Maio
Ano Letivo 2024/2025

Docente: Prof. Miguel Barão

Conteúdo

1 Introdução

A compressão de dados é uma área fundamental na ciência da computação e engenharia, que visa reduzir o tamanho dos arquivos para otimizar o armazenamento e a transmissão de informação. Entre as diversas técnicas de compressão, a compressão baseada na entropia destaca-se por utilizar conceitos da teoria da informação para estimar a quantidade mínima de bits necessária para representar um conjunto de dados sem perda de informação.

Este relatório apresenta a implementação de um programa que calcula a frequência de ocorrência dos caracteres de um ficheiro e estima a sua entropia, que representa o limite teórico para a compressão sem perda possível. A partir dessa análise, é possível compreender melhor a eficiência potencial da compressão de um dado arquivo, fornecendo uma base para o desenvolvimento de algoritmos de compressão mais eficazes.

2 Estrutura do Programa

O programa desenvolvido está estruturado em várias partes principais, organizadas em funções que garantem a modularidade e a clareza do fluxo de execução.

2.1 Entrada do Nome do Ficheiro

Inicialmente, o programa solicita ao utilizador o nome do ficheiro a ser analisado. Para isso, utiliza uma chamada ao sistema para imprimir uma mensagem no ecrã e outra para ler a string introduzida pelo utilizador. A função responsável por esta etapa assegura ainda que a nova linha inserida ao pressionar "Enter" é removida para evitar problemas na abertura do ficheiro.

2.2 Leitura e Contagem dos Caracteres

Após obter o nome do ficheiro, o programa tenta abrir o ficheiro para leitura. Caso a abertura falhe, uma rotina de erro é invocada para informar o utilizador. Se o ficheiro for vazio, uma mensagem de erro correspondente é também apresentada. Caso contrário, o programa lê o ficheiro byte a byte, contabilizando a frequência de ocorrência de cada caractere ASCII válido. Esta contagem é armazenada num array dedicado, permitindo uma análise posterior.

2.3 Cálculo da Entropia

A entropia é calculada com base na fórmula:

$$H(X) = -\sum_{x} p(x) \cdot \log_2 p(x)$$

Como estamos a usar apenas aritmética inteira, esta expressão foi manipulada para evitar a utilização de operações de ponto flutuante. Considerando que:

$$p(x) = \frac{f(x)}{N}$$

o

onde:

- $f(x)$ é a frequência absoluta do caractere x,
- N é o número total de caracteres lidos.

Então, a fórmula da entropia pode ser reescrita como:

$$H(X) = \sum_{x} f(x) \cdot (\log_2 N - \log_2 f(x))$$

Esta expressão permite calcular a entropia total diretamente em bits, utilizando apenas logaritmos inteiros e multiplicações por somas sucessivas. O valor resultante representa o número total de bits necessários para codificar o ficheiro com compressão ideal.

No código Assembly, este cálculo foi feito iterando sobre todas as 128 posições do array de frequências e acumulando o valor:

$$\text{entropia} \mathrel{+}= f(x) \cdot (\log_2 N - \log_2 f(x))$$

2.4 Apresentação dos Resultados

Após o cálculo da entropia total, o valor é arredondado para o byte mais próximo de forma a estimar o tamanho comprimido:

$$\text{Tamanho estimado (bytes)} = \left\lceil \frac{H(X)}{8} \right\rceil$$

Este arredondamento é feito adicionando 8 aos bits e depois dividindo por 8 usando o *srli* que faz um shift right lógico

2.5 Funções de Suporte

Para além das funções principais, o programa conta com várias funções auxiliares que executam tarefas específicas, como a remoção do caractere de nova linha, cálculo do logaritmo de base 2 e multiplicações por somas sucessivas (para evitar o uso direto da instrução de multiplicação).

Esta organização funcional permite que o programa seja claro, modular e fácil de manter, facilitando a sua compreensão e possíveis futuras alterações.

3 Resultados Obtidos

Após a execução do programa com diferentes ficheiros de entrada, foram obtidos os seguintes resultados:

- O programa solicita corretamente o nome do ficheiro e processa o seu conteúdo byte a byte, contabilizando a frequência dos caracteres ASCII presentes.
- Para ficheiros com conteúdo repetitivo, como ficheiros contendo apenas um único caractere repetido, o valor calculado da entropia é muito baixo, aproximando-se de zero, indicando que a compressão sem perda pode ser altamente eficiente neste caso.
- Para ficheiros com diferentes distribuições de caracteres, o valor da entropia calculado reflete a estimativa da quantidade mínima de bits necessários para representar o conteúdo, indicando o limite teórico para a compressão sem perda.
- O programa apresenta mensagens claras e adequadas quando ocorre um erro ao abrir o ficheiro ou quando o ficheiro está vazio, garantindo uma boa experiência ao utilizador.
- A saída apresenta o total de bytes lidos e o valor estimado da compressão, permitindo uma interpretação simples do potencial de compressão do ficheiro analisado.

Estes resultados confirmam a correta implementação dos conceitos de frequência de caracteres e cálculo da entropia, mostrando a utilidade prática do programa para avaliar o limite teórico da compressão de dados sem perda.

4 Dificuldades Encontradas

Ao longo do desenvolvimento do trabalho, surgiram diversos desafios técnicos e de organização que exigiram análise cuidadosa e resolução iterativa. Um dos primeiros obstáculos foi um erro persistente ao tentar carregar dados da memória, que se revelou estar relacionado com a ausência da diretiva .align. Esta diretiva era essencial para garantir o alinhamento correto dos dados, especialmente quando se tratava de aceder ao array das frequências. Este erro também poderia ter sido resolvido, invertendo a ordem na data, colocando a instrução que reserva espaço para *LER* com a instrução que reserva espaço para *FREQA CARACTERES*.

Outra dificuldade importante foi a perda de valores durante a execução de algumas funções. Após investigação, concluiu-se que certos registos estavam a ser sobrescritos ou não eram preservados entre chamadas, o que indicava o não cumprimento do ABI

(Application Binary Interface). Para resolver este problema, foi necessário garantir o restauro adequado dos registos *s** e *ra* na stack, sempre que necessário.

Além disso, houve também o desafio de garantir que todo o código estivesse em conformidade com as boas práticas de organização e modularização. Inicialmente, o código estava muito concentrado na função principal (*main*), o que dificultava a depuração. A posterior reorganização em funções específicas permitiu não só resolver erros como também melhorar a legibilidade e manutenção do código.

No geral, estas dificuldades contribuíram para um maior entendimento das particularidades da programação em Assembly e das exigências de baixo nível na gestão de memória e chamadas de função.

4.1 Diferença encontrada

Após realizar o trabalho (em Windows) testámos como o programa se comportava em Linux. Inicialmente, executou normalmente e corretamente porém, ao testar mais alguns ficheiros, reparámos que lia um caracter a mais do que o suposto. Depois de alguns dias de testes e de tentar perceber a origem deste problema, deparámo-nos com uma pequena diferença entre os editores de texto dos respetivos Sistemas Operativos. Ao utilizar o editor de texto do Linux, é lido o caracter extra mas, se utilizarmos um ficheiro criado e editado com o editor de texto de Windows o programa lê corretamente, o que nos levou a pensar que o possível erro seria que, por defeito, o editor de texto do Linux adiciona um "*newline*" no final do ficheiro.

Utilizámos o VSCode para confirmar a nossa teoria e, de facto, era esse o problema. Ao criar o ficheiro com o VSCode o programa executava e lia a quantidade de caracteres que o mesmo continha mas, ao salvar o programa utilizando o editor de texto do Linux, conseguimos ver uma modificação no ficheiro, e essa modificação era de facto o "*newline*".

Depois de alguns testes, reparámos que também acontece isso ao executar algumas instruções na Prompt de Comando, mas não com todas, por exemplo:

```
printf 'teste > teste1.txt
echo -n 'teste' >> teste2.txt
```

Estas duas instruções criam um ficheiro que não contém o "*newline*" porém ao utilizar a seguinte instrução:

```
echo 'teste' >> teste1.txt
```

o código cria com o caracter "*newline*" o que acaba por dar origem ao erro encontrado.

Estes testes confirmaram que o nosso código não possui erros de contagem de caracteres, o que foi um alívio após diversos dias à procura da origem do erro.

5 Conclusões

Este trabalho permitiu implementar um programa que, baseado na frequência dos caracteres de um ficheiro, calcula uma estimativa da sua entropia e, consequentemente, da sua compressibilidade máxima sem perda. A abordagem utilizada segue os princípios da teoria da informação, permitindo uma análise teórica sólida sobre o limite inferior do tamanho dos dados comprimidos.

Os testes realizados mostraram que a entropia estimada reflete de forma consistente a variedade do conteúdo dos ficheiros: quanto mais repetitivo o conteúdo, menor a entropia e maior a compressibilidade. Por outro lado, conteúdos mais diversificados apresentam maior entropia, indicando menor espaço para compressão.

Além de permitir uma aplicação prática da Teoria da Informação, o desenvolvimento deste programa proporcionou um contacto aprofundado com os desafios típicos da programação em baixo nível, como a gestão de memória, a utilização rigorosa de aritmética inteira e o cumprimento do ABI. O projeto destacou a importância da organização modular do código, da clareza nas funções auxiliares e da robustez no tratamento de casos especiais, já que sendo *Assembly* uma linguagem de baixo nível, devemos facilitar a leitura e interpretação do código.

De forma geral, o trabalho demonstrou de forma eficaz a ligação entre a diversidade do conteúdo de um ficheiro, o valor da sua entropia e o potencial de compressão sem perda. Esta abordagem constitui uma base sólida para compreender os limites teóricos da compressão de dados e poderá servir como ponto de partida para implementações mais avançadas no futuro.

6 Referências

Overleaf - Learn LATEX in 30 minutes
LATEX for Beginners - Workbook
Overleaf - Code listing